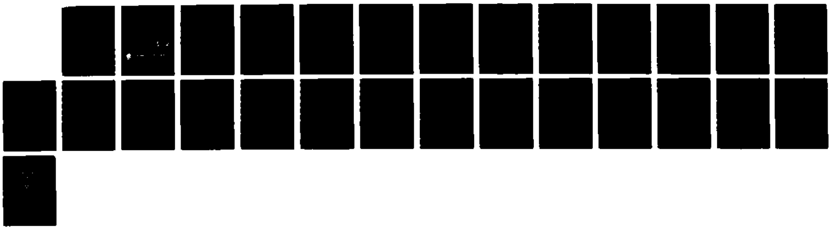
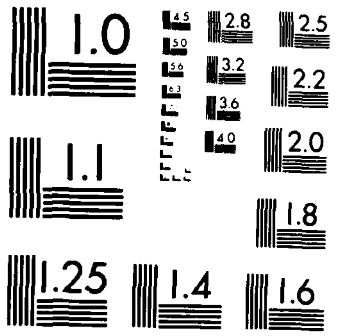
AD-A187 765 VOICE/NATURAL LANGUAGE INTERFACING FOR ROBOTIC CONTROL (U) ARMY ARMAMENT RESEARCH DEVELOPMENT AND ENGINEERING CENTER DOV.. E J CARROLL ET AL. NOV 87 ARFSD-TR-87008 1/1

UNCLASSIFIED F/G 12/9 NL




1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(4)

TECHNICAL REPORT ARFSD-TR-87008

VOICE/NATURAL LANGUAGE INTERFACING FOR ROBOTIC CONTROL

EDWARD J. CARROLL
KEN LAM

NOVEMBER 1987

US ARMY
ARMAMENT MUNITIONS
& CHEMICAL COMMAND
ARMAMENT RDE CENTER

U. S. ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER
FIRE SUPPORT ARMAMENT CENTER
PICATINNY ARSENAL, NEW JERSEY

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER) Technical Report ARFSD-TR-87008	5. MONITORING ORGANIZATION REPORT NUMBER)
6a. NAME OF PERFORMING ORGANIZATION ARDEC, FSAC Fire Control Div — 6b. OFFICE SYMBOL SMCAR-FSF-RC	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (CITY, STATE, AND ZIP CODE) Picatinny Arsenal, NJ 07806-5000	7b. ADDRESS (CITY, STATE, AND ZIP CODE)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION ARDEC, IMD STINFO Br — 8b. OFFICE SYMBOL SMCAR-IMI-I	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (CITY, STATE, AND ZIP CODE) Picatinny Arsenal, NJ 07806-5000	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (INCLUDE SECURITY CLASSIFICATION)
VOICE/NATURAL LANGUAGE INTERFACING FOR ROBOTIC CONTROL

12. PERSONAL AUTHOR(S)
Edward J. Carroll and Ken Lam

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (YEAR, MONTH, DAY)	15. PAGE COUNT
	FROM ______ TO ______	November 1987	30

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (CONTINUE ON REVERSE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)
Natural language Sensor integration Speech recognition
Speech synthesis Voice control Robotic control

19. ABSTRACT (CONTINUE ON REVERSE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

This work examined past and present methods for adapting speech control to computer use. The research resulted in the development of a design methodology for implementation of a preprocessing module to convert from simple, spoken, natural language statements to a computer oriented control language. Specifically, it was designed to take the form of a small, stand alone computer board emulated by an IBM-PC and function as an expert system to translate English statements to a command set appropriate for controlling a Unimation PUMA-560 robotic arm. Even with speech control of the PUMA, the operator was still required to understand the system's capabilities and limitations. However, many of the operating difficulties, resulting from the robot's non-user-friendly programming procedures, were eliminated.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL I. HAZNEDARI	22b. TELEPHONE (INCLUDE AREA CODE) 201-724-3316 — 22c. OFFICE SYMBOL SMCAR-IMI-I

DD FORM 1473, 84 MAR

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

INTRODUCTION

Short, predefined commands have always been the fastest and most accurate way to issue instructions, assuming both parties have the time to memorize and practice the commands. However, communications with a computerized system should be simple and not require the operator to memorize lengthy instructions or even acronym lists. Unfortunately, most system designers and programmers have their own ideas as to what are simple operating procedures; in fact, some commands make sense only to the person who wrote them. In cases where multiple systems and programs must be used by the same person, individual command sequences for each system make learning extremely difficult. Using a valid command, but on the wrong system, is a frequent mistake and very difficult to find. An input processor for these advanced systems, that could extract the meaning from an ordinary English statement and issue the correct commands, would minimize the effects of these common problems. This is the main reason for developing natural language (NL) understanding for computers; not to replace the short, structured commands required to control a system under time critical conditions, but to minimize the training requirement and operator error in all other situations.

EARLY RESEARCH

Voice Synthesis

Systems to generate speech have made great strides within the last few years and have become very popular and easy to integrate into existing systems. This is due mainly to many units featuring standard serial RS232C interfaces and effective text-to-speech algorithms (unplug a standard computer terminal and plug in the synthesizer). Even in the lower priced units, such as ECHO-PC from Street Electronics, an extensive rule base (400+ rules) is used to convert written English to a phoneme string used for controlling the synthesizer. By coupling a built-in text-to-speech algorithm with a top-of-the-line synthesizer, such as Speech Plus' CR-5050 synthesizer, outstanding results can easily be obtained.

Voice Recognition

Computer recognition of the spoken word is still very crude when compared to synthesis. The University of Pennsylvania has excellent on-going research in this area. It presently uses an emulated connection network to recognize individual phonemes as they occur in an input stream of spoken words that it has learned (ref 1). Speech Systems is starting to market a unit that has a 2,000 word vocabulary. This huge vocabulary is not trained by each operator but is accessible by sampling a preselected group of words from the speaker and then looking for phoneme groupings in all other words. However, most systems take a simpler approach and try to match an entire section of an input stream to an item in a stored data base. One variation compresses the data in a decipherable format; an approach taken by the Texas Instruments system for their personal computer. This technique allows reconstruction of the stored data. Other systems, such as the CSRB-VOCALINK by Interstate Voice Products, merely sample and analyze the input steam to arrive at several computed values. Both systems then reference the stored data when deciding which word was spoken. The major problem with this technique is that, due to variations in accents, the equipment must be trained by each user for every word. Unfortunately, until major computing power can be profitably allocated to the speech recognition process, off-the-shelf units will never have sufficient intelligence to function as the main data input or control device.

Natural Language Techniques

Early attempts at NL understanding were mainly limited to a simple pattern match of several words from a sentence. A small matching list or template was used, and if certain words were found, then the sentence meaning was assumed based solely on that match. Famous early programs like ELYZA by Weizenbaum (ref 2) and many front end modules to modern computer games use this technique. The concept is good since much of our recognition of sensory data is a result of a memory search. However, the problems with computers trying to emulate this procedure are small memory and low search speed compared to the human brain. This results in a considerable number of assumptions based on little data.

With greater processing power, the syntax of the input sentence can be examined. In fact, most attempts at NL development use this method. Syntactic parsing uses grammer rules to separate the sentence into units of meaning or groups called phrases. In the early seventies, Winograd used this procedure to program a computer to simulate an imaginary robot operating in the blocks world. After syntactically parsing the sentence (gathering the words into phrase groups), the developed patterns were then matched instead of the individual words as described above. The derived meanings were then used to change information or obtain the answer to a question concerning what the simulated robot was doing or had done (ref 3). The successful implementation of Winograd's program (SHRDLU), stirred many researchers to use and perfect methods for syntactic parsing.

The simplest forms of syntactic parsing are top down, bottom up, and deterministic. Top-down parsing is the most popular. It looks at the sentence as a whole and most often uses an augmented transition network (ATN) to parse by separating the sentences into phrases and then into words. A simple transition network groups phrases by looking for links between words. For example: links for a noun phrase would be between the article, intervening adjectives, and the noun itself. Backtracking and regrouping improperly parsed phrases are practical if the network is augmented with registers. This ATN is the most common method for top-down parsing of natural language statements. Assuming the sentences are grammatically correct, top-down parsing works well. For fragmented sentences, however, regrouping of phrases may go on indefinitely if not stopped due to unrecognizable structures. Spoken English frequently contains imperfect grammer; therefore, bottom-up parsing, which first looks at the words and then builds every possible phrase to form the sentence, is more efficient. Unfortunately with large sentences, many more phrases are grouped than are needed; this takes considerable processing time. There is also deterministic parsing which starts at the front of a sentence and builds the phrase groups as they are encountered. This technique does not allow for backtracking; it looks ahead to attempt correct phrase grouping the first time.

Very few people attempt to analyze the parts of speech while listening to someone talk. At the start of hearing a statement, listeners create a general mental image as the words are received. Additional input serves to reinforce the original image, modify, or change it completely. Emulating this procedure, for anything longer than a short sentence, requires extensive memory searches; this is still not possible to implement in real time with desk top computers.

With a very large NL text to understand, extracting the proper data and making the correct inferences becomes very difficult. Roger Schank, having noticed that people are not overwhelmed by irrelevant inferences but tend to make the right ones at the right times, postulated that people use cultural knowledge in their inferencing process and called this knowledge structure a script (ref 4). Once the correct script is chosen, only specific items of interest need be selected from the text, with the rest of the information accurately assumed.

Many other outstanding artificial intelligence (AI) researchers have also gone beyond simple, straight-forward grammar parsers. The University of California has developed a program that "reads" short politico-economic editorials and answers questions about the editorial contents (ref 5). In the NL lab at the University of Pennsylvania (LINK lab) the researchers have been focusing their attentions on the design of cooperative systems and the computational properties of grammatical formalisms for expressing the syntactic, semantic, and pragmatic meanings of NL utterances (ref 6). However, the greatest percentage of NL research is devoted to accessing data base management systems with very little effort to developing a way for users to access and control computer programs. But despite the popularity of NL processing in the AI community, it has not yet achieved technical credibility or market acceptance (ref 7).

Despite the scarcity of commercially available software for NL understanding, several capable programs exist. Two very well known systems are Language Craft by the Carnegie Group and INTELLECT by Artificial Intelligence Corporation (AIC). Both systems can be configured as required and are reported to perform remarkably well. Language Craft requires the power of a minicomputer or stand alone LISP processor. INTELLECT, an older system, was designed to run on a mainframe; however, it was considered one of the best commercially available systems for NL processing and requires only the development of a custom data base for each application (ref 8).

MODULE EVALUATION SYSTEM

A test bed was conceived for use in validating NL design methodologies (fig. 1). This was to consist of a robotic arm being told, in common English statements, how to move or arrange a group of items on a table. Additionally, a simulated cannon with projectiles (developed in-house for another project) would be used to determine what problems might be encountered by using a natural language module in a simulation of a real system. In this test, the speech system would be used to initiate a process required by a second computer (a VAX/VMS-11/750 from Digital Equipment Corp). This VAX was the primary controller but would rely on data from the voice controlled system to complete its required task.

System Configuration

The robot selected for the testing was a PUMA 560 robotic arm by Unimation, and its NL interface was through an IBM-PC programmed in an interpreted LISP from Integral Quality. LISP was chosen because of the ease in configuring functions which can simulate a custom high-level language. All robot commands generated by this computer would be sent to the PUMA 560 controller through a serial RS232 port operating at 9600 baud. Operator interface to the PC would be by both a standard keyboard and display, and by a voice recognition and synthesis system (CSRB-VOCOLINK and ECHO-PC, respectively). Since the speech recognition system had a limited vocabulary (240 words) and was speaker dependent, keyboard input would be used to supplement the spoken words. However, the synthesizer contained text-to-speech capability and would require no supplemental hardware or software. Finally, a standard NTSC video camera and a Data Translation frame grabber (256 x 240 pixels) would be used to gather data to supplement the robot's position sensors.

Communication Procedures

Several programming choices were available for transferring the robot commands from the PC to the PUMA. The simplest method was to compute each command and send them, one at a time, to the PUMA controller in monitor mode. This method closely emulated a human operator typing at the robot console and allowed any variation of VAL commands to be sent to the robot. However the PUMA controller has a poor input buffer, so the characters in each command had to be sent slowly with each one read and checked for accuracy. This method worked but took too much time for long commands.

The second method involved modifying only individual parameters of preprogrammed commands within an existing robot program and then executing it. Unfortunately, even though fewer characters needed to be sent, there was considerable overhead required for communicating with the robot in the programming mode. This again delayed the actual execution of the desired command.

The third method was to write predefined VAL programs and just execute them by the processor. Since this was workable but allowed little control, it was used in a couple cases but was viewed as useless if full control of the robot by the PC was to be implemented. An additional problem with all three methods was that once each move command was started, it could not be interrupted except by hardware. To alleviate these problems, a memory resident, high level, interactive program was written for the robot controller. This allowed Unimation/VAL-II "signal" commands to interrupt the robot's actions the same way that hardware signals do, and still permitted full use of the hardware interrupts by external sensors. These software signal commands simulated subroutine calls. They were formed by short ASCII strings, such as "signal 1002,1001," a carriage return, followed by any possible data required by the called routine, and finally ended with another carriage return. The PUMA control program constantly looped, checking each signal status, and allowed any move command to be interrupted by any other. This permitted the operator to use a serial port command to alter a previous instruction (programmed as multiple small steps in the desired direction) if a collision appeared eminent. One additional plus is if the robots controller were to be replaced, or a new type of robot arm used, all the required programs to operate the robot would be known. These routines would be proven and ready to code into the new controller. Since the communication between the processors was not VAL dependent, minimal changes in the communication program, in the processor calling the robot, would be required.

PR^GRAM DESIGN CONFIGURATION

The actual natural language program accepted the input steam as defined words which were assumed to contain a valid and complete command in English. The input sentence was then converted to a software image which represented speed, gripper status, and position as related to the PUMA's sensors. This procedure was chosen in an attempt to emulate the mental imaging process people use when listening to a sentence. For example, when someone acts as an interpreter he does not translate word for word between languages. Instead, a mental image, formed from hearing the sentence in the first language, is simply described in the second. The NL interface program was therefore structured as follows:

ENGLISH COMMAND-->INTERMIDIATE SENSORY IMAGE-->TARGET LANGUAGE

The development of this program was simplified by viewing the translation process in reverse. First examined were the capabilities of the PUMA and the commands to perform those actions, then the conversion requirements of a parsed sentence to an intermediate image stage, and finally the grouping of the sentence into units of meaning.

Basic Robot Commands

The VAL-II PUMA control language, like languages for all computerized systems, does not contain ambiguous words; English does. As a result, the domain of discourse (area of interest) was kept narrow so the conversion could be completed within a reasonable amount of time. Fortunately, when the PUMA is operating in its world coordinate system, only three functions need to be addressed: changing the location of the gripper, changing or storing a new coordinate transformation for a location, and opening or closing the gripper's toggles. External to world operations, each joint may be rotated individually; whereas, in the world mode, the joints are automatically controlled so the gripper orientation remains constant to the robot axes.

The subset of VAL-II commands, used for the programs within the PUMA to allow control by an external NL driven processor, is shown in figure 2. Only five of these commands were needed to change the location of the gripper. MOVE permitted the location to be changed to a named location. APPRO told the robot to move the gripper to a spot above the target location in millimeters. DEPART, the converse of APPRO, withdrew the gripper a specific distance above its present location. The speed for executing these instructions was controlled by a simple SPEED command followed by a number to indicate a percentage of the robot's maximum speed.

Within the PUMA controller, locations of objects are stored and addressed symbolically. Therefore, to tell the robot to memorized its present location, the word HERE, followed by a symbolic name, was all that was required. SET duplicated the location transformation or address of a point and applied it to another. Finally, SHIFT allowed a given location address to be offset in millimeters.

Control of the gripper was the last requirement, and its status was remembered by the operator controlling the robot. The commands used for opening and closing the gripper were simply OPEN and CLOSE.

The above VAL-II commands formed the primary instruction set that was required to operate the robot within the confines of the world coordinate system. Additionally, individual joint angles were changed with the DRIVE command.

Control of path planning sensors was not placed under control of the NL module but were polled directly by the robot and interfaced using a hardware parallel port. This port was accessed by REACTI commands (react immediate), which caused the robot to quickly branch within the main control program as lines within its parallel port were pulled true. Two other commands ADC and DAC, used for controlling the analog input and output ports on the robot, were also not placed under operator control but again dedicated to internal robot use. Program control commands such as FOR, IF, CASE, etc., which operate the same way as their counterparts in other computer languages were of great help. These commands, together with computational functions like SIN, ATAN, ABS, etc., were coupled with the parallel and analog input values and used for building intelligence into the robot.

Robot Command Sequences

The next step was to develop a means to select and arrange the VAL commands for position, speed, gripper status, and joint angle. For a simple MOVE, if the destination is not predefined as in the statement "move left," it is computed by using the VAL command DECOMPOSE on a reference or starting location and modifying the values for the x, y, or z axis. Once this axis is altered, a new location is generated using the command TRANS. The robot is then told to move to this new location. After moving there, BITS are used to check if any new "signal" sequence was received from the IBM-PC; if not, the same sequence is repeated. Also, if the initial command from the operator had indicated a continuous motion, such as "move left," the robot would continually repeat the sequence till a new command is issued. Once a new command is received, the robot is halted, its present position is memorized for future reference, and the new command is executed.

Again grouping the motions into more complex sequences, a final set of signal commands were generated. If the robot is instructed to pick up an object, the gripper must depart vertically from the table; to prevent bumping into other items, move to a point directly above the chosen object, then travel straight down to grasp it. That item could then be placed on another spot by reversing the last procedure with new location parameters. Speed was also important not only when moving around or near other objects, but also to limit the stress developed at each joint while transporting a heavy load. The maximum a PUMA can handle at full speed is about 5 lb. These procedures were then coded into a series of signal commands within the PUMA control program and appended to the simpler, primary routines discussed earlier. This approach minimized the number of characters sent from the PC to the robot. Therefore, all required commands to move the robotic arm were coded as signal commands and formed the target language for the translator.

External Sensor Integration

Restricting the instruction domain to the robot's perception of the real world, the PUMA controller was able to perform all the computation required to maintain constant gripper orientation, and the IBM was able to concentrate on language conversion. A video camera was then added to aid in finding selected objects for the robot to access. Before this vision sensor could be used, it needed to be mapped into the robot's workspace so that the visual coordinates translate into robot coordinates. This mapping was done in the PC with all the image processing primitives written in PASCAL to maximize speed. Since voice control of vision processing was not required, the routines to integrate, calibrate, and map the sensor image to the robot's coordinate system, as well as finding the location of a predefined object, were all coded by hand and simply executed when needed. These procedures are described in the appendix.

Language Translation

Conversion from the natural language input to the intermediate concept state had to be completed before the correct signal commands could be selected (fig. 3). As explained earlier, there were several ways this could be accomplished. Since the commands were to be spoken and since the whole processing power of an IBM PC was dedicated to understanding English, a variation of bottom-up parsing was selected.

Statement Rewrite

The first step taken to understand a sentence was to simplify the structure by rewriting the statement in an easier to understand format. This was done by looking for complex word groups, and replacing these groups with a simpler word, or phrase that means the same thing, such as replacing the compound preposition "on top of" to simply "on." Also, words may be eliminated if they are superfluous and used to embellish sentences either for social reasons (e.g., please) or just to sound impressive.

Pattern Matching

The pattern matching procedure is very straightforward. The input sentence is first formed into a list (L1). Then, a second list (L2), which contains sublists of patterns and the replacements, is matched against the first. L2 has the following structure:

[((pattern1)replacement1)((pattern2)replacememnt2)...((patternN)replacementN)].

The first sublist from L2, ((pattern1)replacement1), is initially removed. The first sublist from that sublist, (pattern1), is then compared, one word at a time to the front of L1. If no match is made then the first word of L1 is removed and the matching is attempted again. After all words are removed from L1, they are then reassembled; the second main sublist from L1 is extracted, and the matching process is repeated. This process is repeated until L2 is exhausted. If a pattern matched, then the matched phrase is replaced in L1 by the corresponding replacement from L2. The matching then continues with the same pattern until L1 is exhausted. The process is deliberate and slow, but since the domain of discourse was limited, this matching served to speed the parsing process. For expanded domains, other matching techniques may be required. If done efficiently, matching might be used to entirely replace the syntactic parsing procedure and still extract all the information from a sentence. For this procedure to be practical, however, memory size and processing speed must both be greatly increased.

Phrase Grouping

The simplified sentence was parsed based on a subset of grammer rules. All the words were grouped to find all possible phrases. Simple noun phrases (NP), which consist of any number of adjectives followed by a noun, may be preceded by a preposition to form a prepositional phrase (PP). An NP may also contain a PP as a modifier. An example would be the NP "the large book on top of the white table." "The white table" forms a simple NP (NP1). The compound preposition "on top of" forms a PP with NP1 as it object. "The large book" forms NP2 which is modified by the PP. Verb phrases (VP) that may be formed by verbs and adverbs or adverbial phrases were also parsed. Conjunctions, like "and," form a dual idea and may double a location or action or even split the entire sentence to form two individual concepts. Far more complicated structures exist in the English language but should not occur for this type of application.

The individual phrases were then examined by following rules to find their meanings. The VPs described the actions and formed the address or name of a function to evaluate. The NPs contained information pointing to location names; the nouns referring to a general group of locations with the adjectives and modifying phrases narrowing the selection. The prepositions were used to indicate an offset from the evaluated locations. The subject of the action determined whether the robot was to move or just modify location transformations. Conversion from this format to signal commands was then completed by rules specific to the type of application.

For the initial parsing, a dictionary, which contained a description of each word, was all tnat was required. These words and definitions formed sublists within one major list. For example:

[(a (article indefinite))... (zebra (noun...))].

Because of this list formation, adding additional words was a simple matter; adding more rules, however, was far more complex since an understanding of "what must be done" is required before it can be determined "how to do it." As a result, additional information for computer use had to be presented by the operator and not simply derived by the computer. If a computer system had full control of external sensors, the sensor data might be able to be used as a source of knowledge to help the computer evaluate a changing environment, and ultimately add new rules to adapt for these changes. Before this can occur, development of a separate module is required that can use sensor drivers and primitives to gather data in a intelligent manner. For this project, a data base of phrases was used as a simple updatable method for the system to learn other means of verbally representing its knowledge (ref 9).

Restatement

Following the phrase parsing and identifying stages, a rule base was accessed to decide how the sentence could be restated in the target language. Appending additional rules to this data base enabled the system to understand, or at least respond to, new methods of describing actions to the robot. The final output from this stage was also a list. The first element pointed to a function which took the remainder of the list as its' data. When evaluated, this function output the proper signal commands to the robot to carry out the action requested by the operator.

MODULE DESIGN EVALUATION

Test Design

All words that might be needed for the test demonstration were grouped. The operator then spoke each word into the voice recognition system to train the unit to his voice. Since a carriage return was needed to signal the system that a complete sentence was input, the system was trained to insert a carriage return (Ctrl-M) when hearing the word "enter;" the word "now" was also permitted to allow for smoother sounding commands. The robot was then verbally instructed to move its gripper in different directions, pick up objects, and then placed them down or on top of other objects. Finally, the robot was told to insert an aluminum shell into a simulated cannon when a command to load was received from the VAX computer.

Results

Generally, if all the spoken words were accurately recognized, the correct signal commands were generated. However, the complete parsing procedure took between ten seconds to over a minute, especially for longer sentences or poorly phrased statements. Since the code was operating in interpreted mode, this was not suprising; if the program was compiled, it could be expected to run over ten times faster. An even greater speed increase would be possible if the code were run in a system that was specifically designed to execute LISP commands. Since the NL translation worked well for this test, a small section of the developed NL design module was recoded in complied PASCAL (the same language used to control the robot's vision) in a effort to maximize the speed of the translation without using a LISP machine. For simple robot commands, the conversion process was reduced to real time. This simplified the testing of the coordination and control of the voice recognition system, the voice synthesis system, the vision sensor, and the robot itself.

The voice synthesizer messages were initially difficult to understand. Therefore, the ECHO-PC was replaced with the CR-5050, which resulted in dramatic improvement.

The only real problem encountered was with voice recognition. The system used was fully programmable, but in the normal operating mode, there was a significant difference in the recognition accuracy between testing the unit while alone in the lab and while demonstrating it before visitors. In the latter case, due to slight changes in voice quality apparently from nervousness, the correct word selection occurred only about 60% of the time. When the operator was alone and relaxed the recognition rate was over 90%.

CONCLUSIONS

Voice/natural language control of robotic systems is definitely possible; however, there are problems.

1. Unless a very large computer is available and allowed to dedicate most of its processing time to NL understanding and access to a data base is the only requirement, there is very little software available for the researcher.

2. An adaptable voice recognition system must be used to ensure accuracy during stressful times. In a combat situation, the user will be under considerable stress so the voice recognizer must adapt, perhaps not for different users but at least for the same person, whether he is tense or at ease.

3. Fixed routines for sensor operation are too limiting. For efficient operator use of sensors, each system should be able to be verbally "taught" how to use them to simplify integration and operation.

4. To ensure efficiency, system communication must be minimized by requiring each sensor to process its own data with all cognitive processes, such as NL understanding, relegated to a single supervisory computer.

RECOMMENDATIONS

Since bit level manipulations are required for direct sensor control, dedication of a large computer to handle natural language translations, as well as all required transformations for each sensor, would be too demanding on one system. This would also produce a communication nightmare with volumes of raw data being transferred between computers. Therefore, built-in intelligence for each sensor is necessary if real-time natural language control of sensor-based systems is to be realized. With this low level built-in intelligence, a top level supervisory computer would only be required to instruct the control processors on how to modify and coordinate their actions based on English commands. These processors would each consits of a low level computationally oriented computer, with each performing its own individual task of either controlling motors (robotic positioning) or processing sensor data. Single board computers might be adequate for simple tasks, but for more complicated procedures, specialized processors or even parallel architecture systems like connection machines (ref 10) or neural networks (ref 11) might prove to be the best choice. Above these modules, a high level symbolic oriented computer would supervise the processors and make decisions based on knowledge it has accured either from preprogrammed data, past history, or from information derived from polling its sensors. Since Texas Instruments is now supplying LISP chip sets, and Symbolics just announced their single chip LISP processor, small inbedable symbolic oriented processors are now available and are no longer limited to laboratory use. A standardized high level intersystem communication protocol (the target language for the NL conversion module) will need to be developed. This language must be capable of handling commands to and from all required positioning devices and sensors. Very high speed would not be necessary since each module will have its own built-in intelligence, only transfer of processed data and minimal operational instructions will be required. Applying this approach to system design should allow the creation of a full voice/natural language environment. This system could then replace or enhance current push button, dial, and display environments which are currently the only method of control available to robotic system operators.

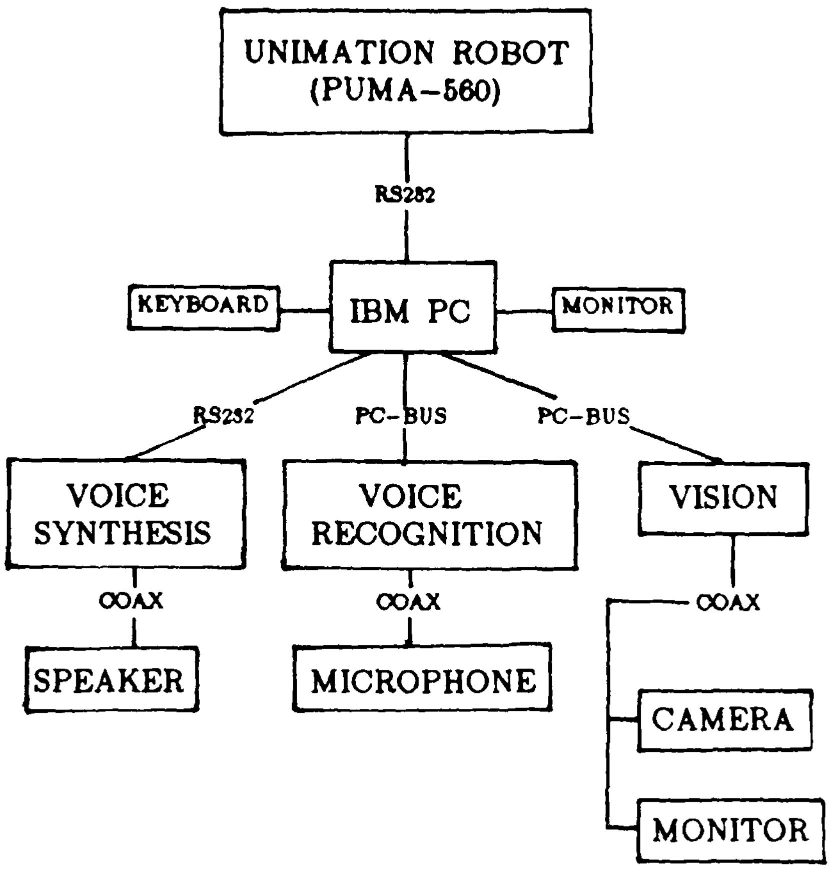


Figure 1. Development and evaluation system

POSITION CONTROL

CHANGING LOCATIONS	NEW COORDINATE TRANSFORMATIONS	GRIPPER CONTROL
	WORLD AND JOINT MODE	
MOVE	HERE	OPEN
APPRO	SET	CLOSE
DEPART	SHIFT	
SPEED		
	JOINT MODE	
DRIVE		

DATA AND PROGRAM CONTROL

FOR,IF,etc

DAC,ADC

REACTI

SIG

SIGNAL

ABS,SIN,etc

Figure 2. Robot command subset

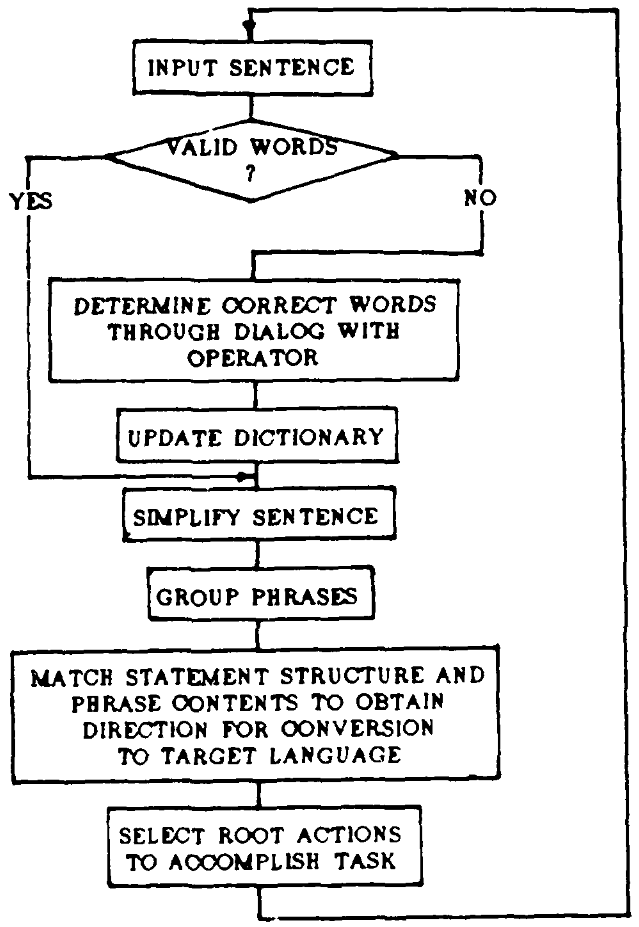

INPUT SENTENCE
VALID WORDS ?
YES
NO
DETERMINE CORRECT WORDS THROUGH DIALOG WITH OPERATOR
UPDATE DICTIONARY
SIMPLIFY SENTENCE
GROUP PHRASES
MATCH STATEMENT STRUCTURE AND PHRASE CONTENTS TO OBTAIN DIRECTION FOR CONVERSION TO TARGET LANGUAGE
SELECT ROOT ACTIONS TO ACCOMPLISH TASK

Figure 3. Natural language module design

REFERENCES

1. Shastri, Lokendra and Watrous, Raymond L., "Learned Phonetic Discrimination Using Connectionist Networks," Technical Report, University of Pennsylvania, Philadelphia, PA, 1986.

2. Hassmer, Tony, Looking at LISP, Addison-Wesley, 1984.

3. Tennent, Harry, Natural Language Processing, Petrocelli Books, 1981.

4. Dyer, Michael G., "Understanding Computer Understanding of Narrative Text," Mathematical Social Sciences, vol. 6, pp 353-395, 1983.

5. Alverado, Sergio J; Dyer, Michael G; and Flowers, Margot, "Editorial Comprehension in OpEd Through Argument Units," Proceedings of the Fifth National Conference on Artificial Intelligence, August 1986.

6. Cheikes, Brient A., "Research in Artificial Intelligence at the University of Pennsylvania," The AI Magazine, pp 128-143, August 1986.

7. Obermeier, Klaus K., "Natural Language Processing - Facts and Figures," Proceedings of the Third Annual Artificial Intelligence & Advanced Computer Technology *Conference*, pp 419-424, April 1987.

8. Andrews, Mark, "The Four Faces Of AI," Easy Home Computer, January 1984.

9. Zernik, Uri and Dyer, Michael G., "Towards a Self-Extending Lexicon," Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, July 1985.

10. Hillis, W. Daniel, "The Connection Machine," Scientific American, June 1987.

11. Hartley, Karen, "Seeing the Need for 'Art'," Science News, vol 132, 1987.

APPENDIX

VISION/ROBOT MAPPING (PASCAL)

The orientation and scaling between the robot's and camera's coordinate systems was computed first. The aspect ratio of the camera, CamAspRatio (a fixed value determined by the camera manufacturer), was obtained from the camera by measurement and found to be 1.24 pixels vertical/horizontal. Two markers placed at known locations on the table by the robot within the field of view of the camera were used as calibration points. The PC determined the visual centers of the markers from a grabbed frame and the actual centers, in robot coordinates, from the PUMA controller. The visually measured centers of the two markers ([X1v,Y1v] and [X2v,Y2v]) and the robot measured centers ([X1r,Y1t] and [X2r,Y2r]) were then used to determine the scaling (Sc1Axs), rotation (RotAng), and translation (TransX and TransY) values for converting from vision to robot position coordinates.

The visual origin was shifted and centered at the first marker. Therefore, the origin coordinates were then computed, in PASCAL.

```
XOrgVis := X1v*CamAspRatio

YOrgVis := Y1v
```

Likewise, the translation values were determined.

```
TransX := X1r

TransY := Y1r
```

The rotation angle was then computed.

```
RotAng := ArcTan ((Y2r - Y1r)/(X2r - Xlr)) - ArcTan ((-(Y2v - YOrgVis))/

            (X2V*CamAspRatio - XorigVis))
```

The scaling value (the same for both axes since the aspect ratio was measured eariler) was then derived.

```
Sc1Axs := Sqrt(Sqr(X2r - X1r) + Sqr(Y2r - Y1r))/Sqrt(Sqr(X2v*CamAspRatio -

            XorigVis) + Sqr(-(Y2v - YOrgVis)))
```

After the translation, scaling, and rotation factors were computed, the vision sensor was then able to be used for positioning the robot. The visually measured coordinates of points ([Xv,Yv]) and rotation angles (ObjAng) were converted to the robot coordinates and angles. First, the camera aspect ratio and the new origin were used to scale and compute the new visual coordinates from the measured frame grabber coordinates.

Xv := Xv*CamAspRatio - XOrigVis

Yv := -(Yv - YorgVis)

The robot coordinates for the point (RbtPtX,RbtPtY) were then computed.

RbtPtX = ((Xv*Sc1Axs)*Cos(RotAng) - (Yv*Sc1Axs)*Sin(RotAng)) + TransX

RbtPtY = ((Xv*Sc1Axs)*Sin(RotAng) - (Yv*Sc1Axs)*Cos(RotAng)) + TransY

Finally, the ture angle in the robot system (RbtPtAng) was determined.

RbtPtAng : = ObjAng - RotAng

As with the primitives, all higher level operations involving vision were hand coded for use by the NL module. These operations were also written in PASCAL and involved the development of the following procedures:

Quickly finding objects by intensities

Finding object edges within defined spaces

Determining angles of edges

Screen drawing routines, such as bordering the object of interest and sketching the object, by gray level intensities, on the PC screen.

DISTRIBUTION LIST

Commander
Armament Research, Development and
 Engineering Center
U.S. Army Armament, Munitions and
 Chemical Command
ATTN: SMCAR-IMI-I (5)
 SMCAR-CC, H. Opat
 SMCAR-FS, H. Garver
 T. Davidson
 SMCAR-FSA, R. Wrenn
 SMCAR-FSF-R, L. Ambrosini
 J. Lehman
 J. Lester
 N. Coleman
 E. Carroll (5)
 L. Lam (5)
 SMCAR-FSA-E, F. Saxe
 SMCAR-TDC, H. Grunder
Picatinny Arsenal, NJ 07806-5000

Commander
U.S. Army Armament, Munitions and
 Chemical Command
ATTN: AMSMC-GCL (D)
Picatinny Arsenal, NJ 07806-5000

Administrator
Defense Technical Information Center
ATTN: Accessions Division (12)
Cameron Station
Alexandria, VA 22304-6145

Director
U.S. Army Materiel Systems Analysis Activity
ATTN: AMXSY-MP
Aberdeen Proving Ground, MD 21005-5066

Commander
Chemical Research, Development and
Engineering Center
U.S. Army Armament, Munitions and
Chemical Command
ATTN: SMCCR-MSI
Aberdeen Proving Ground, MD 21010-5423

Commander
Chemical Research, Development and
Engineering Center
U.S. Army Armament, Munitions and
Chemical Command
ATTN: SMCCR-RSP-A
Aberdeen Proving Ground, MD 21010-5423

Director
Ballistic Research Laboratory
ATTN: AMXBR-OD-ST
Aberdeen Proving Ground, MD 21005-5066

Chief
Benet Weapons Laboratory, CCAC
Armament Research, Development and
Engineering Center
U.S. Army Armament, Munitions and
Chemical Command
ATTN: SMCAR-CCB-TL
Watervliet, NY 12189-5000

Commander
U.S. Army Armament, Munitions and
Chemical Command
ATTN: SMCAR-ESP-L
Rock Island, IL 61299-6000

Director
U.S. Army TRADOC Systems
 Analysis Activity
ATTN: ATAA-SL
White Sands Missile Range, NM 88002

Project Manager
Ammunition Logistics
ATTN: AMCPM-AL, G. Kent
Picatinny Arsenal, NJ 07806-5000